

## Ice Sheet System model User interface

Eric LAROUR<sup>1</sup>, Eric RIGNOT<sup>1,3</sup>, Mathieu MORLIGHEM<sup>1,2</sup>, Hélène SEROUSSI<sup>1,2</sup> Chris BORSTAD<sup>1</sup>, Feras HABBAL<sup>1,3</sup>, Daria HALKIDES<sup>1,4</sup>, Behnaz KHAKBAZ<sup>1</sup>, John SCHIERMEIER<sup>1</sup>, Nicole SCHLEGEL<sup>1</sup>

<sup>1</sup>Jet Propulsion Laboratory - California Institute of Technology

<sup>2</sup>Laboratoire MSSMat, École Centrale Paris, France

<sup>3</sup>University of California, Irvine

<sup>4</sup>Joint Institute for Regional Earth System Science & Engineering, UCLA



[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

## Outline

① startup

② Model class

③ Plots

④ Mesh

EXP files  
triangle

⑤ Parameterization

Interpolation modules  
SeaRISE example

⑥ Mask

⑦ Flow equations

⑧ Diagnostic

Diagnostic parameters  
Boundary conditions

⑨ Prognostic

⑩ Transient 2D

Transient solution

⑪ Extrusion

⑫ Thermal

⑬ Transient 3D

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Startup file

- `startup.m` must be present when matlab is launched
- Automatically executed by matlab at “start up” to load all ISSM tools

```
1 %Recover ISSM_TIER and USERNAME
2 ISSM_TIER=getenv('ISSM_TIER');
3 USERNAME =getenv('USER');
4 if (isempty(ISSM_TIER)),
5     error('issmdir error message: ''ISSM_TIER'' environment variable is empty! You should ...
       define ISSM_TIER in your .cshrc or .bashrc!');
6 end
7
8 %Now add all issm code paths necessary to run issm smoothly.
9 %We capture the error output, so that we can warn the user to update
10 %the variable ISSM_TIER in this file, in case it is not correctly setup.
11
12 %ISSM path
13 addpath([ISSM_TIER '/src/m/utils/']); %loads recursivepath
14 addpath([ISSM_TIER '/doc']);
15 addpath([ISSM_TIER '/bin']);
16 addpath(recursivepath([ISSM_TIER '/src/m']));
17 addpath(recursivepath([ISSM_TIER '/externalpackages/scotch']));
18 addpath(recursivepath([ISSM_TIER '/externalpackages/canos']));
19 addpath(recursivepath([ISSM_TIER '/externalpackages/kml']));
20 addpath(recursivepath([ISSM_TIER '/externalpackages/googleearthtoolbox/']));
21 addpath(recursivepath([ISSM_TIER '/externalpackages/export_fig']));
```

[User interface](#)

Larour et al.

startup

## Model class

Plots

Mesh

EXP files

triangle

## Parameterization

Interpolation modules

SeaRISE example

Mask

Flow equations

## Diagnostic

Diagnostic parameters

Boundary conditions

## Prognostic

## Transient 2D

Transient solution

## Extrusion

## Thermal

## Transient 3D



# Matlab Model class

```

1  >> md=model
2
3  md =
4
5      mesh: [lxl mesh]           -- mesh properties
6      mask: [lxl mask]          -- defines grounded and floating elements
7      geometry: [lxl geometry]  -- surface elevation, bedrock topography, ice ...
8      thickness, ...
9      constants: [lxl constants] -- physical constants
10     surfaceforcings: [lxl surfaceforcings] -- surface forcings
11     basalforcings: [lxl basalforcings] -- bed forcings
12     materials: [lxl materials] -- material properties
13     friction: [lxl friction] -- basal friction/drag properties
14     flowequation: [lxl flowequation] -- flow equations
15     timesteppping: [lxl timesteppping] -- time stepping for transient models
16     initialization: [lxl initialization] -- initial guess/state
17     rifts: [lxl rifts] -- rifts properties
18     debug: [lxl debug] -- debugging tools (valgrind, gprof)
19     verbose: [lxl verbose] -- verbosity level in solve
20     settings: [lxl settings] -- settings properties
21     solver: [lxl solver] -- PETSc options for each solution
22     cluster: [lxl none] -- cluster parameters (number of cpus...)
23     balancethickness: [lxl balancethickness] -- parameters for balancethickness solution
24     diagnostic: [lxl diagnostic] -- parameters for diagnostic solution
25     groundingline: [lxl groundingline] -- parameters for groundingline solution
26     hydrology: [lxl hydrology] -- parameters for hydrology solution
27     prognostic: [lxl prognostic] -- parameters for prognostic solution
28     thermal: [lxl thermal] -- parameters for thermal solution
29     steadystate: [lxl steadystate] -- parameters for steadystate solution
30     transient: [lxl transient] -- parameters for transient solution
31     autodiff: [lxl autodiff] -- automatic differentiation parameters
32     flaim: [lxl flaim] -- flaim parameters
33     inversion: [lxl inversion] -- parameters for inverse methods
34     qmu: [lxl qmu] -- dakota properties
35     results: [lxl struct] -- model results
36     radaroverlay: [lxl radaroverlay] -- radar image for plot overlay
37     miscellaneous: [lxl miscellaneous] -- miscellaneous fields

```



[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Model fields

- Model properties are sorted by type:
  - mesh properties in `md.mesh`
  - material properties in `md.material`
  - ...
- Each model field is itself a matlab object with fields
- Default parameters are provided (physical constants, etc)

```

1  >> md.materials
2
3  ans =
4
5  Materials:
6
7  rho_ice           : 917          -- ice density [kg/m^3]
8  rho_water          : 1023         -- water density [kg/m^3]
9  mu_water           : 0.001787    -- water viscosity [N s/m^2]
10 heatcapacity        : 2093         -- heat capacity [J/kg/K]
11 thermalconductivity : 2.4          -- ice thermal conductivity [W/m/K]
12 meltingpoint        : 273.15       -- melting point of ice at latm in K
13 latentheat          : 334000       -- latent heat of fusion [J/m^3]
14 beta                : 9.8e-08     -- rate of change of melting point with pressure [K/Pa]
15 mixed_layer_capacity : 3974         -- mixed layer capacity [W/kg/K]
16 thermal_exchange_vel...: 0.0001     -- thermal exchange velocity [m/s]
17 rheology_B           : N/A          -- flow law parameter [Pa/s^(1/n)]
18 rheology_n            : N/A          -- Glen's flow law exponent
19 rheology_law          : 'Paterson'   -- law for the temperature dependance of the ...
               rheology: 'None', 'Paterson' or 'Arrhenius'
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

## Saving and loading a model

- md is a placeholder for all parameters, constants, geometrical properties, etc.
  - No additional file required, everything is in ONE matlab variable.
  - Use matlab's `save` command to save a model in binary format
  - Use `loadmodel` to reload a model from a binary file
- makes sure that old models are loaded correctly

```
1 >> md=model;
2 >> md.miscellaneous.name
3
4 ans =
5 ..
6
7 >> md.miscellaneous.name='test';
8 >> md.miscellaneous.name
9
10 ans =
11     'test'
12
13 >> save myfirstmodel md
14 >> loadmodel md;
15 >> md.miscellaneous.name
16
17 ans =
18     'test'
```

[User interface](#)

Larour et al.

[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Display model fields

- `plotmodel` can be used to display any model field/result
- Arguments (pairs of options):

- 1 model
- 2 'data'
- 3 name of model field
- 4 name of option1
- 5 option1 value
- 6 name of option2
- 7 option2 value
- 8 ...

```
1 >> plotmodel(md, 'data', md.mesh.x);
2 >> plotmodel(md, 'data', md.mesh.x, 'colorbar', 0, 'caxis', [0 500000]);
3 >> plotmodel(md, 'data', 'mesh');
4 >> plotmodel(md, 'data', 'BC');
5 >> plotmodel(md, 'BC', 'data', md.mesh.x, 'data', 'mesh');
```

- More info:

- `plotdoc`
- <http://issm.jpl.nasa.gov/documentation/usersmanual>

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

## EXP format

- We rely on Argus' format for geometrical files
- Argus files have an `exp` extension
- Example for a square domain:

```
1 ## Name:domainoutline
2 ## Icon:0
3 # Points Count  Value
4 5 1.
5 # X pos Y pos
6 0 0
7 100000 0
8 100000 100000
9 0 100000
10 0 0
```

- ISSM has tools to read/write/modify `exp` files
- This format is used for:
  - domain outline (mesh boundary)
  - floating ice extension
  - flag elements or vertices inside a profile

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Mesh generation with triangle

- triangle is used to generate simple uniform unstructured meshes

- Arguments:

- 1 model
- 2 name (string) of the domain outline
- 3 element mean size

```
1 md=model;
2 md=triangle(md, 'Square.exp', 10000);
```

- To display the mesh:

```
3 plotmodel(md, 'data', 'mesh');
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Parameterization file

- The parameter file sets up all required fields of a model:
  - Geometrical properties (Upper surface elevation, thickness,...)
  - Boundary conditions (friction, front,...)
  - Solver settings
  - ...
- Example for a square ice shelf:

```
1 hmin=300;
2 hmax=1000;
3 ymin=min(md.mesh.y);
4 ymax=max(md.mesh.y);
5 md.geometry.thickness=hmax+(hmin-hmax)*(md.mesh.y-ymin)/(ymax-ymin);
6 md.geometry.bed=-md.materials.rho_ice/md.materials.rho_water*md.geometry.thickness;
7 md.geometry.surface=md.geometry.bed+md.geometry.thickness;
8
9 pos=find(md.mask.vertexonfloatingice);
10 md.friction.coefficient=200*ones(md.mesh.numberofvertices,1);
11 md.friction.coefficient(pos)=0;
12 md.friction.p=ones(md.mesh.numberofelements,1);
13 md.friction.q=ones(md.mesh.numberofelements,1);
14
15 md.initialization.vx=zeros(md.mesh.numberofvertices,1);
16 md.initialization.vy=zeros(md.mesh.numberofvertices,1);
17 md.initialization.vz=zeros(md.mesh.numberofvertices,1);
18 md.initialization.vel=zeros(md.mesh.numberofvertices,1);
19
20 md.materials.rheology_B=paterson((273-20)*ones(md.mesh.numberofvertices,1));
21 md.materials.rheology_n=3*ones(md.mesh.numberofelements,1);
22
23 md=SetIceShelfBC(md,'Front.exp');
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Interpolation modules

- Interpolation from a regular grid to a mesh: `InterpFromGridToMesh`

```
md.initialization.temperature=InterpFromGridToMesh(x,y,temp,md.mesh.x,md.mesh.y,250);
```

- Interpolation from between two meshes: `InterpFromMeshToMesh2d`

```
md.initialization.temperature=InterpFromMeshToMesh2d(index,x,y,temp,md.mesh.x,md.mesh.y);
```

## User interface

Larour et al.

startup

Model class

Plots

Mesh

EXP files

triangle

Parameterization

Interpolation modules

SeaRISE example

Mask

Flow equations

Diagnostic

Diagnostic parameters

Boundary conditions

Prognostic

Transient 2D

Transient solution

Extrusion

Thermal

Transient 3D



## SeaRISE example

```

1 #####Greenland parameters#####
2 md.miscellaneous.name='SeaRISEgreenland';
3
4 modeldatapath = [issmdir '/projects/ModelData/SeaRISE/Greenland5km_v1.2'];
5 thicknesspath =[modeldatapath '/thk.mat'];
6 surfacepath =[modeldatapath '/usrf.mat'];
7 bedrockpath =[modeldatapath '/topg.mat'];
8 vxpath =[modeldatapath '/surfvelx.mat'];
9 vypath =[modeldatapath '/surfvely.mat'];
10 temperaturepath =[modeldatapath '/surftemp.mat'];
11 precippath =[modeldatapath '/smb.mat'];
12
13 #####Some hardcoded parameters for this deck#####
14 disp('      reading geometry');
15 md.geometry.thickness=InterpFromFile(md.mesh.x,md.mesh.y,thicknesspath,0);
16 pos=find(md.geometry.thickness>1);
17 md.geometry.thickness(pos)=1;
18 md.geometry.surface=InterpFromFile(md.mesh.x,md.mesh.y,surfacepath,0);
19 md.geometry.bathymetry=InterpFromFile(md.mesh.x,md.mesh.y,bedrockpath,0);
20 md.geometry.bed=md.geometry.surface-md.geometry.thickness;
21
22 disp('      reading velocities ');
23 md.inversion.vx_obs=InterpFromFile(md.mesh.x,md.mesh.y,vxpath,0);
24 md.inversion.vy_obs=InterpFromFile(md.mesh.x,md.mesh.y,vypath,0);
25 md.inversion.vel_obs=sqrt(md.inversion.vx_obs.^2+md.inversion.vy_obs.^2);
26 md.initialization.vx=md.inversion.vx_obs;
27 md.initialization.vy=md.inversion.vy_obs;
28 md.initialization.vz=zeros(md.mesh.numberofvertices,1);
29 md.initialization.vel=md.inversion.vel_obs;
30
31 disp('      creating friction parameters');
32 md.friction.p=ones(md.mesh.numberofelements,1);
33 md.friction.q=ones(md.mesh.numberofelements,1);
34 md.friction.coefficient=30*ones(md.mesh.numberofvertices,1);
35 min_drag_coeff=35; background_drag_coeff=200;
36 md.friction.coefficient=background_drag_coeff*ones(md.mesh.numberofvertices,1);
37 pos=find(md.inversion.vel_obs>30);
38 md.friction.coefficient(pos)=background_drag_coeff+(min_drag_coeff-background_drag_coeff)/maxvel*md.inversion.vel_obs;
39
40 disp('      loading temperature ');
41 md.initialization.temperature=InterpFromFile(md.mesh.x,md.mesh.y,temperaturepath,0)+273.15;
42 md.initialization.pressure=md.materials.rho_ice*md.constants.g*(md.geometry.surface-md.mesh.z);
43 ...

```



[User interface](#)

Larour et al.

[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Mask

- `setmask` is used to generate areas where ice is grounded and floating
- **Arguments:**
  - ① model
  - ② floating ice domain
  - ③ grounded ice inside the floating ice
- Ice considered grounded by default
- Input files in Argus format
- Examples

`md=setmask(md,"")` → all grounded

`md=setmask(md,'all','')` → all floating

`md=setmask(md,'IceShelves.exp','')` → grounded with some floating parts

`md=setmask(md,'all','Islands.exp')` → floating with some grounded parts

- To display the mask:

```
1 >> plotmodel(md, 'data', md.mask.elementonfloatingice)
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Flow equation

`setflowequation` is used to generate the approximation used to compute the velocity

- Arguments:

- ① model
- ② approximation names
- ③ approximation domains

- Domains can be Argus files or array of element flags

- Approximation available

- stokes (Full-Stokes model)
- pattyn (Higher-order model)
- macayeal (Shallow Shelf Approximation)
- hutter (Shallow Ice Approximation)

- Model coupling possible (see tomorrow's presentation)

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Flow equation

`setflowequation` is used to generate the approximation used to compute the velocity

- Examples

```
1 md=setflowequation(md,'hutter','all')
2 md=setflowequation(md,'stokes','all')
3 md=setflowequation(md,'macayeal',md.mask.elementonfloatingice,'pattyn',md.mask.elementongrounded
4 md=setflowequation(md,'macayeal','IceShelves.exp','fill','pattyn')
```

- To display the type of approximation:

```
1 >> plotmodel(md,'data','elements_type')
```

## User interface

Larour et al.

Mesh

EXP files

## triangle

## Interpolation modules

© 2007-2010

### Diagnostic parameters

### Transient solution

## Diagnostic parameters

Most diagnostic parameters can be found in `md.diagnostic`.

Launch diagnostic solution with:

```
>> md=solve(md,DiagnosticSolutionEnum)
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Boundary conditions

- » md=SetIceSheetBC (md)

→ Dirichlet BC for all nodes on boundary

- » md=SetIceShelf (md, 'Front.exp')

→ Neumann BC for all nodes on boundary in 'Front.exp'

→ Dirichlet BC for all other nodes on boundary

- » md=SetMarineIceSheefBC (md)

→ Dirichlet BC for all nodes on grounded boundary

→ Neumann BC for all nodes on floating boundary

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Prognostic parameters

Most prognostic parameters can be found in `md.prognostic`

```
1  >> md.prognostic
2
3  ans =
4
5  Prognostic solution parameters:
6      spcthickness          : N/A      -- thickness constraints (NaN means no constraint)
7      hydrostatic_adjustment : 'Absolute' -- adjustment of ice shelves surface and bed ...
8          elevations: 'Incremental' or 'Absolute'
9      stabilization          : 1        -- 0->no, 1->artificial_diffusivity, ...
10         3->discontinuous Galerkin
11
12  Penalty options:
13      penalty_factor         : 3        -- offset used by penalties: penalty = Kmax*10^offset
14      vertex_pairing          : N/A      -- pairs of vertices that are penalized
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Prognostic parameters

**Use `md.timestepping` to change the time step:**

```
1  >> md.timestepping
2
3  ans =
4
5  timestepping parameters:
6      time_step           : 0.5      -- length of time steps [yrs]
7      final_time          : 5        -- final time to stop the simulation [yrs]
8      time_adapt          : 0        -- use cfl condition to define time step ? (0 or 1)
9      cfl_coefficient     : 0.5      -- coefficient applied to cfl condition
```

**Launch prognostic solution with:**

```
1  >> md=solve(md,PrognosticSolutionEnum)
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Transient solution

Most transient parameters can be found in `md.transient`

```
1  >> md.transient
2
3  ans =
4
5      transient solution parameters:
6          isprognostic      : 1      -- indicates if a prognostic solution is used in the ...
7              transient
8          isthermal         : 1      -- indicates if a thermal solution is used in the transient
9          isdiagnostic       : 1      -- indicates if a diagnostic solution is used in the ...
10             transient
11          isgroundingline   : 0      -- indicates if a groundingline migration is used in ...
12              the transient
13          requested_outputs : N/A    -- list of additional outputs requested
```

Transient solutions in 2D combine:

- diagnostic
- prognostic
- grounding line

→ Some of these components can be deactivated

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Transient solution

**Use `md.timestepping` to change the time step:**

```
1  >> md.timestepping
2
3  ans =
4
5  timestepping parameters:
6      time_step           : 0.5      -- length of time steps [yrs]
7      final_time          : 5        -- final time to stop the simulation [yrs]
8      time_adapt          : 0        -- use cfl condition to define time step ? (0 or 1)
9      cfl_coefficient     : 0.5      -- coefficient applied to cfl condition
```

**Launch transient solution with:**

```
1  >> md=solve(md, TransientSolutionEnum)
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Extrusion

- extrude is used to extrude the 2d mesh into a 3d mesh

- Arguments:

- ① model
- ② number of layers
- ③ lower extrusion exponent
- ④ upper extrusion exponent (optional)

- Examples

```
1 md=extrude(md,8,1)
2 md=extrude(md,10,1.5)
3 md=extrude(md,10,1.5,1.5)
```

- To display the mesh:

```
1 plotmodel(md,'data','mesh');
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

## Thermal solution

Most thermal parameters can be found in `md.thermal`

```
1  >> md.thermal
2
3  ans =
4
5      Thermal solution parameters:
6          spctemperature      : N/A           -- temperature constraints (NaN means no ...
7              constraint)
8          stabilization       : 1            -- 0->no, 1->artificial_diffusivity, 2->SUPG
9          maxiter             : 100          -- maximum number of non linear iterations
10         penalty_lock        : 0            -- stabilize unstable thermal constraints ...
11             that keep zigzagging after n iteration (default is 0, no stabilization)
12         penalty_threshold    : 0            -- threshold to declare convergence of ...
13             thermal solution (default is 0)
```

Thermal solution in 3D only to compute

- thermal steady state
  - thermal transient
- Controlled by `md.timestepping.timestep`

Launch thermal solution with:

```
1  >> md=solve(md,ThermalSolutionEnum)
```

[User interface](#)[Larour et al.](#)[startup](#)[Model class](#)[Plots](#)[Mesh](#)[EXP files](#)[triangle](#)[Parameterization](#)[Interpolation modules](#)[SeaRISE example](#)[Mask](#)[Flow equations](#)[Diagnostic](#)[Diagnostic parameters](#)[Boundary conditions](#)[Prognostic](#)[Transient 2D](#)[Transient solution](#)[Extrusion](#)[Thermal](#)[Transient 3D](#)

# Transient solution

Most transient parameters can be found in `md.transient`

```
1  >> md.transient
2
3  ans =
4
5      transient solution parameters:
6      isprognostic          : 1      -- indicates if a prognostic solution is used in the ...
7          transient
8      isthermal             : 1      -- indicates if a thermal solution is used in the transient
9      isdiagnostic           : 1      -- indicates if a diagnostic solution is used in the ...
10         transient
11      isgroundingline       : 0      -- indicates if a groundingline migration is used in ...
12          the transient
13      requested_outputs     : N/A   -- list of additional outputs requested
```

Transient solutions in 3D combine:

- thermal
- diagnostic
- prognostic
- grounding line

→ Some of these components can be deactivated

A wide-angle photograph of a desolate, icy terrain. In the foreground, a flat expanse of white, textured snow or ice stretches across the frame. Behind it, a range of mountains rises, their peaks covered in thick, white snow. The mountains are rugged, with deep shadows in the valleys and bright reflections on the snow. The sky above is a clear, pale blue, with no clouds visible.

Thanks!